

METHOD AND SYSTEM FOR EFFICIENTLY STORING WEB PAGES FOR QUICK DOWNLOADING AT A REMOTE DEVICE

Technical Field

5 This invention relates to methods and systems for efficiently storing web pages at a server for quick downloading at a remote device in communication with the server.

Background Art

10 Traditional web servers depend on sequential access to HTML (HyperText Markup Language) pages by loading hyperlinked pages as the client requests them while navigating through the current page. Each client initiated hit involves a random access to the memory, even if multiple clients access the same page. Thus, overall latency is high even when there is an overlap in the access. Latency refers to the amount of time it takes a web server to respond to a client
15 beginning from the time the request was sent by the client to the server. To reduce latency in the network, some proposals have been made to have local machines, mostly proxy servers, to prefetch the pages linked to the current page before the client actually "hits" them. However, since all web pages may not be requested by the client, storage is wasted. Furthermore, bandwidth is also wasted. If a user is
20 on a public area network, the user will pay for the time of access or number of bytes transferred every time he/she uses the bandwidth. Consequently, if the network is busy, aggressive prefetching can be expensive.

25 File servers, on the other hand, typically utilize a hierarchical storage manager (HSM) in managing access of data stored on disk and tape. An application program in an HSM configuration examines memory usage in the hard drive. Instead of putting all files, or data, on the hard drive, the program will keep a subset of the files on the hard drive and the rest on the tape. By ensuring that the most currently or frequently used files are on the hard drive, most accesses are satisfied by accesses to the hard drive, i.e., hits.

If the file required is not on the hard drive, i.e., a miss, a memory request is sent to the tape. Since the tape is slower than the disk, the time to retrieve the file is higher. Since the access is usually to the disk, this penalty is only apparent on a small percentage of the accesses. Thus, the main motivation for using
5 HSM is cost. Tape is significantly cheaper and denser than a hard drive. However, in some file requests, latency will suffer.

Thus, there exists a need for a method and system for efficiently storing scaleable amounts of data at a web server that can be quickly accessed by a client without compromising bandwidth.

10

Disclosure Of The Invention

It is a general object of the present invention to provide a method and system for efficiently storing web pages at a server for quick downloading at a remote device in communication with the server.

15

In carrying out the above object and other objects, features, and advantages, in one embodiment of the present invention a method is provided for efficiently storing web pages for quick downloading at a remote device in a computer network including a server computer having a first storage device for storing a first plurality of web pages, a second storage device for storing a second
20 plurality of web pages linked to the first plurality of web pages, and a third storage device coupled to the computer for storing a third plurality of web pages linked to the first plurality of web pages. The method includes receiving a first signal from the remote device at the server computer indicating selection of one of the first plurality of home pages. The method also includes transferring at least one of the
25 second plurality of web pages linked to the selected one of the first plurality of web pages from the second storage device to the first storage device in response to the first signal. Still further, the method includes transmitting a second signal from the second storage device to the third storage device in response to the first signal. The method also includes transferring at least one of the third plurality of web pages
30 linked to the selected one of the first plurality of web pages from the third storage

device to the second storage device in response to the second signal wherein anticipated web pages linked to the selected one of the first plurality of web pages are quickly accessible by the remote device.

In further carrying out the above object and other objects, features,
5 and advantages, in another embodiment of the present invention, a web server is provided for carrying out the above described method. The web server includes a first memory for storing a first plurality of web pages and for receiving a first signal from the remote device indicating selection of one of the plurality of web pages. The web server also includes a second storage device for storing a second plurality
10 of web pages linked to the first plurality of web pages and for transferring at least one of the second plurality of web pages linked to the selected one of the first plurality of web pages to the first storage device in response to the first signal. The second storage device is further provided for transmitting a second signal in response to the first signal. The web server further includes a third storage device for
15 storing a third plurality of web pages linked to the first plurality of web pages, receiving the second signal, and transferring at least one of the third plurality of web pages linked to the selected one of the first plurality of web pages to the second memory in response to the second signal wherein anticipated web pages linked to the selected one of the first plurality of web pages are quickly accessible by the re-
20 mote device.

The above object and other objects, features and advantages of the present invention are readily apparent from the following detailed description of the best mode for carrying out the invention when taken in connection with the accompanying drawings.

Brief Description Of The Drawings

FIGURE 1 is a schematic diagram of the storage-efficient web server of the present invention;

FIGURE 2 is a directed page graph for a page sequence that begins
5 with page i;

FIGURE 3 is an example of a page distribution in the server storage hierarchy; and

FIGURE 4 is a flow diagram illustrating the general sequence of steps associated with the present invention.

Best Modes For Carrying Out The Invention

A schematic diagram of the storage-efficient web server is shown in Figure 1, denoted generally by reference numeral 10. The server 10 includes a computer 12 such as, for example, a workstation or a high-end personal computer. The server 10 hosts a plurality of web sites, wherein each web site includes a
15 plurality of HTML documents, or web pages.

The computer 12 of the server 10 includes a fast memory 14, such as cache or RAM (random access memory), and a disk memory 16. Although shown externally, the disk memory 16 may be either an internal disk drive or an externally attached disk or disk farm, such as a RAID (redundant array of inex-
20 pensive disks). Both the fast memory 14 and the disk memory 16 store a portion of the web pages for quick downloading as will be described in greater detail below.

The computer 12 is also connected to an external storage device 18, such as tape or removable media, which stores even more data relating to the web pages. The storage device 18 can be connected to the computer 12 either directly
25 via a point-to-point link, as shown in Figure 1, or indirectly via a storage area

network. The server 10 is in communication with a plurality of remote devices 20 which can download any of the web pages stored at the server 10. Examples of such devices include a computer, an Internet-ready telephone, etc.

As shown in Figure 1, a three-tier hierarchy exists; fast memory 14,
5 disk memory 16 and storage device 18, where the traditional assumption of lower access speed, larger capacity, and lower cost holds true as one moves down the hierarchy. While in most cases these may represent cache or RAM, disk and tape, any other forms of storage that satisfies the access speed, capacity and cost
10 monotonicity also applies. In any case, the storage hierarchy, which can be extended to more than three tiers, forms a serial buffer chain where the cache in fast memory 14 is supported by a cache in disk memory 16 which in turn is supported by the data in the storage device 18.

Also, this chained buffer configuration allows the possibility of data to be transferred between fast memory 14 and disk memory 16 while storage device
15 18 transfers data to disk memory 16, provided disk memory 16 utilizes a typical double buffer so that one buffer feeds disk memory 16 while the second buffer is filled by storage device 18. This simultaneous data transfer between hierarchies is possible as long as disk memory 16 possesses adequate data bandwidth and connectivity either through direct point-to-point links or through a storage area
20 network.

The server 10 of the present invention reduces cache space at fast memory 14 storage level which results in fewer pages of a web site being put on the computer 12 while off-loading successive pages in the second level storage of the storage device 18. Before continuing with the description of the present invention,
25 an exemplary directed page graph is shown in Figure 2 for a page sequence that begins with page i. Note that because the page numbering convention is visit order independent, the naming sequence is not unique. Therefore, many different page graphs can result in the same page number sequence.

Returning to the discussion of the present invention, in order to reduce the cache space at the fast memory 14, a large number of the first level home pages, i.e., the first page of the web sites, i, j, k , etc. are stored in the fast memory 14, as shown in Figure 3. Disk memory 16 contains a larger sequence of subsequent levels of the web site pages, i.e., i, i_{11} and i_{12} , where i_{jk} refers to the k th hyperlinked page on the j th level.

First level home pages, e.g., i , are kept in fast memory 14 while the pages i_{11} and i_{12} , hyperlinked by i , are kept in disk memory 16. The cache in fast memory 14 has links to locations on disk memory 16, as shown by the connections in Figure 3. That is, when cached page i is accessed from fast memory 14, a pointer points to the location (address) of the same page i in disk memory 16. Similarly, the sequence for pages i, i_{11}, i_{12} in disk memory 16 reference the address in storage device 18 so that on further access (beyond i_{12} in disk memory 16), the server 10 knows where to fetch the rest of the pages from storage device 18.

When there is a hit on i in fast memory 14, all pages linked to i in disk memory 16 are read into fast memory 14 even before there is a hit on any of i_{11} or i_{12} . Therefore, while the page is sent to the client, fast memory 14 reads in i_{11} and i_{12} , according to the example shown in Figure 2. Thus, prefetching of certain pages from disk memory 16 reduces the total number of home pages that normally resides in fast memory 14. By using cache space reuse, described below, the space in fast memory 14 can be kept relatively small for each web site hosted.

Cache space reuse implies that the cache used in fast memory 14 is kept to a fixed maximum per home page. This is achieved by prefetching from disk memory 16, discarding pages not required in the cache for fast memory 14 and reading in pages from storage device 18.

To follow on the simple example described above with reference to Figures 2 and 3, the following sequence is executed, as shown in Figure 4. In the following example, it is assumed that disk memory 16 stores only the first level of web pages linked to the home page, while storage device 18 stores the remaining

levels of web pages linked to the home page. The storage space of disk memory 16 actually depends on the access time of locating partitions in storage device 18, i.e., finding the address in storage device 18 where the pages are written. For example, an address utilized in a disk-type storage device 18 is identified by a {sector, track, block number} descriptor. If storage device 18 is very fast, then either only the first level or the first and second level pages only could be stored in disk memory 16. A further explanation of the storage space of disk memory 16 is described in greater detail below.

In continuing with the example, a first signal is transmitted from a client at one of the remote devices 20 requesting one of the home pages, e.g., i , as shown at block 100. The server computer 12 receives the first signal, as shown at block 110.

In response to the first signal, disk memory 16 transfers the first level web pages linked to the selected home page, i.e., i_{11} and i_{12} , to fast memory 14, as shown at block 112. At the same time, disk memory 16 transmits a second signal to storage device 18 requesting specific data, as shown at block 114. In response to the second signal, storage device 18 transfers the second level web pages linked to the selected home page to disk memory 16, i.e., i_{21} , i_{22} and i_{23} , as shown at block 116.

If the client requests one of the first level web pages, e.g., i_{12} , a third signal is transmitted from the remote device 20 accordingly, as shown at block 118. Fast memory 14 then deletes the remaining non-selected first level web pages, i.e., page i_{11} , as shown at block 120, and sends the selected web page to the client. Since disk memory 16 has read in and stored pages i_{21} , i_{22} , i_{23} , etc., from storage device 18, disk memory 16 transfers the second level web pages related to page i_{12} , i.e., page i_{23} , to fast memory 14. Similarly, disk memory 16 may delete the non-selected second level web pages if cache memory size is a concern. Then, disk memory 16 transmits a fourth signal to storage device 18 requesting the web pages linked to the selected page i_{23} , i.e., pages i_{33} and i_{34} . The method continues in a like manner as the client requests additional web pages.

It is possible that the client may proceed to traverse the web pages in a reverse order back to a previous web page where there were more than one possible path from that web page, e.g., page i utilizing the example shown in Figure 2. In this instance, upon reaching that web page, disk memory 16 transmits a signal to storage device 18 requesting the web pages linked to that page, e.g. pages i_{11} and i_{12} , as described at block 112. Thus, disk memory 16 anticipates the client's request by insuring fast memory 14 always has all of the first level web pages related to the selected web page stored therein. At the same time, disk memory 16 insures that all of the second level web pages related to the selected web page are stored in disk memory. This is achieved via memory re-use as described above.

The above sequence depends on the access time of locating partitions in storage device 18, wherein partitions correspond to the portion of the storage device 18 that comprises all the web pages and sequences of the associated web site. If the time to access a partition in storage device 18 is t_{M2} , then the number of pages to be kept in M1 will be dependent on two factors: i) average size, $s(i_{jk})$, of an individual page i_{jk} at any level j in the page directed graph, and ii) reaction time of the client, assumed fixed t_r . The level j of the number of pages to be hosted in disk memory 16 is determined according to the following:

$$j \geq \sum_j t_p + t_r + \frac{s(i_{jk})}{L} \geq t_{M2}$$

where t_p is the average propagation delay between client and server, L is the average link bandwidth, or effective data rate, between the client and server 10, and $s(i_{jk})$ over k represents the average size of the pages at level j .

A simple explanation of the limit to j is as follows. Each level of the tree traversed (i.e., $j=1, 2$, etc.) provides the server 10 the following time window consisting of the client reaction time, t_r , which is probably 0.25 seconds for the average human, the delay or propagation time, t_p , and the time of flight $s(i_{jk})/L$ to send the average page at level j to the client to access storage device 18. As the average page size $s(i_{jk})$ increases, its time of flight provides a larger time window

during which access to a slow storage device 18 can be masked due to the limited link bandwidth L . The propagation delay is typically ignored since the client can be at any arbitrary distance from the server 10.

5 The above inequality is somewhat conservative since it assumes that the average page size is a good indicator of the time of flight. In the case where there is a large variation in file size, the mean should be replaced with a minimum, $\text{Min}(s(i_{jk}))$. This results in increasing the depth of the pages to be stored and potentially increases the storage required in disk memory 16.

10 An example illustrating the use of the above limit is as follows. If the link bandwidth L to the server 10 is 56 Kbs, the average page size at each level in the page directed graph of the home page sequence is 50 KB, and the access time to storage device 18 is eight seconds, then at least two levels of pages ($j=2$) must be resident in disk memory 16. In case of the i page sequence, pages i , $i11$, $i12$, $i21$, $i22$ and $i23$ must be kept in disk memory 16.

15 Typically, however, the average page loaded on the Web is only 6 KB. This implies that a larger depth (level) of pages may be stored in disk memory 16 for a fixed size cache. However, since the pages are of smaller size, the total storage required (i.e., number of pages per level*average size per page*number of levels) will not be significantly different.

20 To ensure that successive pages can be read from storage device 18 with no perceived latency, a quasi-sequential access of the levels is utilized, i.e., page $i_{j+1,l}$ is always read after page $i_{j,k}$ even when $k < l$. If $i_{j,k} = i11$, then $i_{j+1,l} = i2l$, where l is any integer 1, 2, 3, . . ., etc. Access to web site pages is not totally random since only certain paths can be traversed by a client, nor is the access totally
25 sequential since each web site typically has multiple yet a finite number of links on each level. By exploiting this quasi-sequentiality, slow sequential access storage devices can be exploited.

5 All home page sequences are added at a boundary of the partition so that the partition can be accessed with moderately low deterministic access time. Vertical partitioning works well in this case, i.e., data stored sequentially. If access time increases to further out partitions, one can place home pages with very large pages at the further ends and home pages with small pages at the beginning of the device 18, i.e., vertical partitioning. In vertical partitioning, each partition starts from the same starting point and proceeds from left to right so that the access time to the first page on any partition is zero whenever the tape is read.

10 Clearly, storage devices 18 that exhibit low partition access latency will be well-suited to developing hierarchical storage web servers as disclosed herein with very large capacities and with the added benefit of fully masking latencies to disk memory 16 and storage device 18.

15 However, the storage device 18 may be a non-sequential device, such as, for example, a random access optical disk. In this case, the present invention is still valid even though partitioning as described above is not necessary.

While the best modes for carrying out the invention have been described in detail, those familiar with the art to which this invention relates will recognize various alternative designs and embodiments for practicing the invention as defined by the following claims.